**Evaluation**

**Introduction**

Through this Evaluation I will explain my choices of languages, libraries and programs used within my system. I will consider the user interface, database design and queries used throughout my program.

As my programming language of choice, I have decided to use Python, as it is the language I am most comfortable with and since I had already spent a lot of time learning to code in python, spending time to learn another language would severely hinder the production of my solution. Taking these into consideration, I felt that it would be my best option to opt for Python as my language of choice.

With respect to the libraries I used for the system, I used a Tkinter framework made for Python. Having experience with Tkinter, it was the best option in terms of creating UI for my system.

My database of choice is **DB Browser for SQLite**. Like Tkinter, I have had experience with SQL and so I was familiar with its syntax and how to incorporate it into my system. I was able to create CRUD functionalities using SQL queries used in my program. To gain access to SQLite in my code, I had to use the sqlite3 library in Python. Additionally, I had to update my version of SQLite to gain access to additional data manipulation queries that weren’t previously available in the older versions of SQLite.

**Tools and Techniques used**

In Python, I was able to design my program using an object-orientated program (OOP). The reason for this approach was that I needed a system where a user could switch between multiple views and OOP was suitable for these needs. Overall, Python is a high-level programming language, which means that learning new coding techniques was made easier since it closely resembles the English Language.

Additionally, Python is an interpreted language which means that code is executed line by line. Therefore, if an error is met within my program, the whole system would stop and give a reference to the location of the error. This made pinpointing errors within my code easier when developing and testing my system.

To help with detecting errors in my code, I decided to use the **Visual Studio Code editor** downloaded on my Laptop instead of using the school’s Python IDE as VS Code had more functionalities that would help with the development of my system. These included several extensions such as Syntax highlighting, Formatting and Debugging tools which I could download easily from VS Code’s built-in library of extensions.

**GitHub Repository**

Since I was working at home and in school on my project, I needed an easier way to store my program. From my experience with coding on past projects, I decided to use GitHub, which is a platform for hosting code that allows for version control. This means that if I ever needed to revert to a previous version of my system, I could do so through pulling past save files from GitHub. GitHub also allows me to write messages when I commit a version of my program. Therefore, using this I could quickly pick up from where I left off by reading these messages. Throughout the course of this project I also had to learn how to use GitHub to make saving my work my efficient and time effective.

A screenshot of a computer

Description automatically generated

Whilst I had initially planned to design my system using an OOP approach, it still took me a while to understand, which resulted in the development process being delayed towards the beginning of the solution.

Additionally, I was advised to use the MVC (Model, View, Controller) approach for designing my system. This would allow me to store the data from SQLite in a Model, store the UI of my system as separate Views, and link them together through a Controller.

During the early stages of using this approach, I was met with lots of Cyclical errors in my code, since I was incorrectly implementing the MVC method. After this experience, I decide to do research into other ways of designing my system. This is where I came across **Singletons.** The idea behind this was to create an instance of my Views (a singleton) and store them in a **ViewManager.**

A screen shot of a computer program

Description automatically generated**A screenshot of a computer program

Description automatically generatedViewManager**

**A screen shot of a computer program

Description automatically generated**

From here I could access all my views from 1 file and thus choose what to display/hide/register etc.

This method of implementing Singletons into my system was very challenging, as I had to change the way I access data from my Database and used it in my system. In the future I would like to use Singletons from the beginning of the development process to give me more time to adjust to the complexity of the system.

**SQL Controller**

I decided to implement my CRUD functionality a little differently in my system. Instead of importing the sqlite3 library into every single view I had, I created a file which would take SQL queries and execute them in one file.A screen shot of a computer

Description automatically generated

To do this, each view would have a Controller connected to it, which would have to access to number of services. These services would be responsible for passing the SQL queries into the SQLController. This would mean that I would only have to import the SQLController into each of the services. This allowed for my code to become more reusable, as if I wanted to add a new query, I could just add it to an existing or a new service.

**Comparing with Existing solutions**

Similarly to my investigation chapter, there are few existing solutions of Lesson Managers that I have access to. This lack of examples meant that designing my own system would be much harder, as I had few variations of the existing solutions to choose from

From analysing these existing solutions, I have realised that my solution will not be accessible because of the use of the Tkinter library. The swimming teachers require the Lesson Manager to be in a browser, not a downloadable file. If I could choose a different programming method to make this system I would choose the Django framework, so that I could run my program on a web server. However, since I knew more about the Tkinter library, it was the more suitable choice for this project.

When comparing my system with these other solutions, I found that I took more inspiration from the “MorLeisure” system than the “SwimSoft” one.

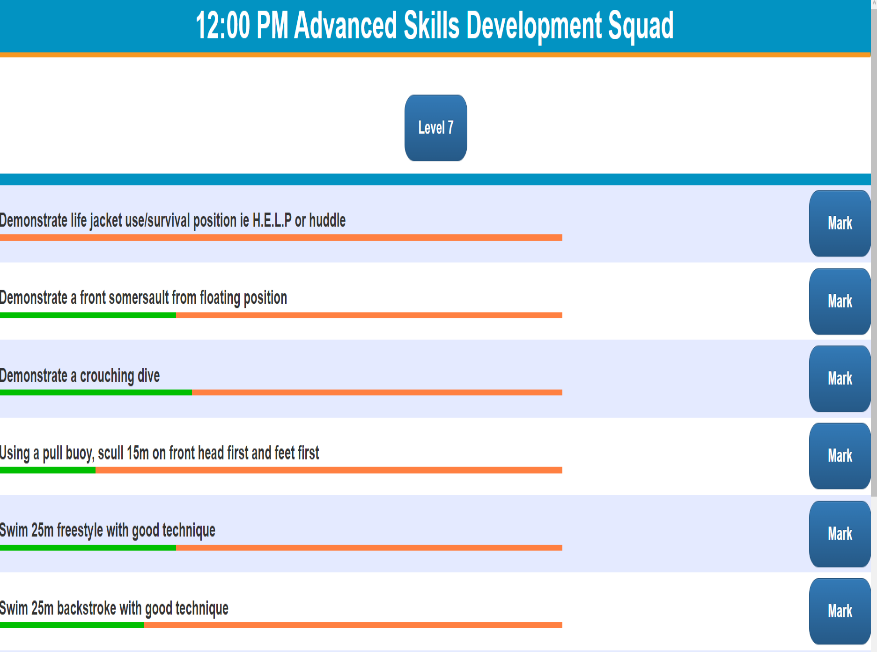
**A screenshot of a computer

Description automatically generatedMorLeisure Class Select**

This was the model that I took inspiration from. I have adapted it to better cater towards the user requirements of my swimming club. From my Class Select you can view classes on multiple days rather than just the current day you are on. This is a big improvement to the system and is an example of how it improves upon the “MorLeisure” solution.

A functionality that I would have liked to implement into my system was to display the number of swimmers in each class through the register button. This is not essential for my program and couldn’t develop it due to time constraints.

Overall a successful adaptation of the existing software.

**MorLeisure Assessing Swimmers**

Another comparison to make between the 2 systems is the Assessment View. Viewing the progress of swimmers in the existing solution is shown via the progress bar that becomes greener as the swimmers are passing in their skills. This is a more user friendly solution that I wish I could have implement into my program.

I decided not to implement this type of feature as it was beyond the scope of my program, and thus wasn’t required for my system. Alternatively, I tried to introduce colour highlighting into my Assessment View, where if a swimmer had passed their skill, it would be highlighted in green. This proved to be too complex for me to design and so I had to scrap this idea too. Overall a good attempt to recreate the user experience from the existing solution.

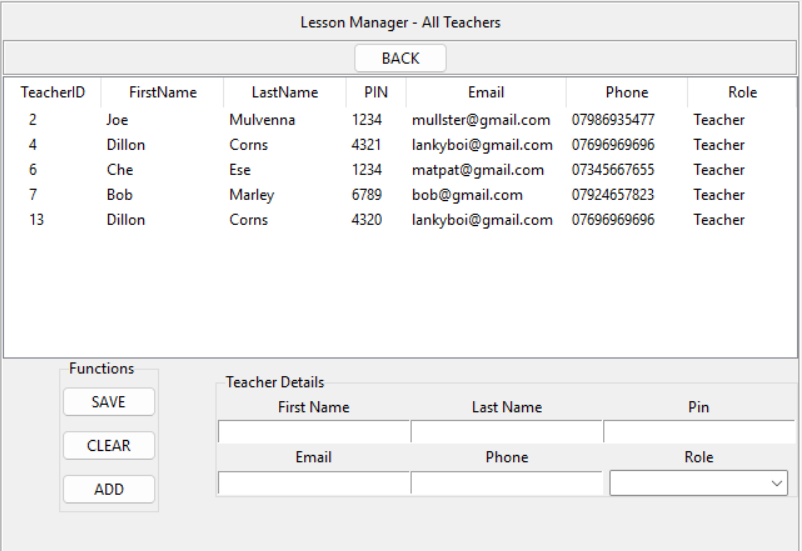
**Good Features and Shortcomings**

The user interface I have provided is quite simple and relatively intuitive to use. The text provided for each function and colours used within the system helps to give the user an idea of what everything does.

One design choice I would change would be the **Staff View.** My design involved showing all staff as 2 separate roles: Teachers and Assistants. This could have been simplified to just displaying all Staff in one big treeview and explaining their role within the club, but again this was an oversight.

A screenshot of a computer

Description automatically generated**All Teachers View All Assistants View**

(This could have been merged into 1 treeview for reusability)

A screenshot of a computer

Description automatically generatedAs a whole, I feel like the All Classes View is definitely one of the more well put functionalities within the system, as it provides the manager with the ability to ADD/REMOVE/UPDATE classes.

The addition of Labelframes to display the purpose of different widgets within the frame allows for newer users to easily interpret the function of certain buttons and class details. For example, without the “Class Details” Labelframe, there would be no way to tell what the purpose of the ComboBoxes were in the system.

Another aspect of how the user interface could be improved is how the overall design and creativity of the UI is presented to the user. For example, the “Edit Class” view doesn’t contain label frames to explain the purpose of various widgets within the frame, and so this function seems less intuitive. On top of this, all the widgets are packed closely together, which could lead to a bad user experience for newer users.

A screenshot of a computer

Description automatically generated

Lots of white space in the SOW section, which takes up more room in the view.

An example of how Labelframes could have been implemented into this view.

For future projects, I would like to try to incorporate more of styled approach to allow for a better user experience. For instance, more colour, highlighting and sectioning widgets into different frames.

User input confirmation also allows for a more friendly user experience. This was implemented into “Edit SOW View”. The “Edit SOW View” is a new addition to the system as in previous existing solutions, the SOW had to be edited manually through the database which wasted time and wasn’t as intuitive.

**A screenshot of a computer

Description automatically generatedEdit SOW View**

**A screenshot of a computer

Description automatically generated**

This design is simple and is very intuitive to use, as there is very little functionality that needs to be explained here to a new user. You simply, click on the desired SOW, a text widget appears, you make amendments and click SAVE.

Another shortcoming of this system is that each user cannot select how to view all Classes. For example, the user would be able to select: **All Lessons, All My Lessons, Current Lesson** after logging into the system. This would further enhance the user experience as the system would have to recognise who is logging into the system, and would have to select classes depending on the users choice. However, this was purely a quality-of-life feature and so due to time constraints, I had to stop development for this since it wasn’t something that would prevent the whole system from working.

**Strengths and Weaknesses**

Creating the initial design for the system, my plan was to make all my code reusable so that in the long run of the project, I would spend very little time creating new views and creating queries for fetching data from the database. This plan however, had taken a large amount of time to fully implement and as such, certain functionalities in the system had to be omitted, such as removing a swimmer from a class or sending parents a swimmer’s report via email. Had I done this project again, I would set more realistic expectations of what I can accomplish within a certain timeframe. Overall, my program has been designed in such a way that it would be easy now to create these additional functions, so the reusability of my code would be a **Strength**, however, the size and scope of the project I would classify as a **Weakness.**

With respect to user requirements for the system, I have been able to complete most of the requests from the Staff at the swimming club. Certain functionalities couldn’t be implemented due to their complexity and as such it resulted in those functions being omitted from the system. In doing this, I had completely disregarded communicating this to the Staff at the swimming club, so moving forward, I would set realistic expectations for the system instead of trying to create functions for the system that are way above my knowledge in Python. This I would consider a **Weakness.**

By using the Visual Studio Code editor, I was able to fully take advantage of its built-in functions, such as the Debug tool. Using this allowed me to identify errors in my code quickly and fix them efficiently too. I definitely think that choosing this IDLE greatly benefitted me in the making of this system, as without it I wouldn’t have been able to detect problems as effectively. This is a **Strength** of the development for this system

Overall, I feel that the time spent on the user experience for the system was not prioritised as much as the functionality of the system. This led to a user experience which could have been better if more time had been dedicated to the user interface. Additionally, making the UI on the system more spaced out is what I would change if I were to do this project again, as it resulted in lots of confusion with newer users of the system when testing. This is a **Weakness** of the system.

**Review**

This project was completed using the “Waterfall” method. This allowed for a lot more documentation of the system than I thought was required. Although this method is useful for other programmers trying to grasp the idea of my system, I felt that it was too much time spent on write-ups.

If I were to redo this project, I would use the “Agile” method, as this allows for a more dynamic programming style that would allow for a better programming experience when creating the system. This approach would also cater to the needs of the user, as I would be able to go back and change certain features that user would want or not want anymore. This would also end up with a more accurate system that the user wants. With the “Waterfall” method, in the time spent between investigating the user requirements and creating the system, the user could change there preferences completely, which would lead to a solution which is not meeting the user’s needs.

**Investigation**

My investigation chapter as I had stated before was too ambitious and wasn’t accurately representative of my programming abilities. Going into a new project, I would make sure that I would be sure of implementing all functionalities of the system before trying to design anything. This would save hours of wasted time trying to solve problems that could have been spent on another area of the system.

Additionally, I would have liked to be more closely in contact with the users of the system, as it would have allowed me to change the system as it is being created, to suit their requirements.

**Design**

I am quite happy with my design chapter, however, there were major design flaws when implementing blueprints for the system. This resulted in me completely changing the design of certain diagrams to be able to implement them more affectively. Additionally, I would have liked to include more detail in some of my diagrams, as the lack of detail led to inaccurately designed functions, which would have to be changed later. This would have saved a lot of time in the long run.

I also could have changed my design chapter after deciding to omit or completely change certain features from the system. For instance, in my Data Dictionary for my design chapter, I realised that I wouldn’t need a Registers table for my classes, since I had access data from my database using each class\_id instead of a register\_id. Similarly, I wouldn’t have needed a Level table for the system, as I realised that I could store the level details inside the SOW instead.

Finally, I felt that the pseudocode in my Design chapter was way too simple for the complexity of the system, and so I feel in the future I shouldn’t try to spend time to write them up.

**Prototype**

The prototype of my system was a very good start to my project. It had all of the key requirements that was asked for in the system, aside from a few quality of life features, such as Sorting Classes. Validation wasn’t a part of my Prototype as I felt it wasn’tfortesting purposes.

**Software Development**

My system operates as planned and has no major flaws which would prevent it from functioning as intended. I feel that doing that Software Development chapter would have been better if I had completed it alongside other chapters, as I naturally had thoughts about my program that would have been insightful in this chapter. Since I started the Software Development after I completed the rest of my chapters, it meant that I had to go back to refamiliarize myself with all the previous plans I had made, which wasted time.

**Testing**

My test chapter as it stands could provide more content than what is currently given. Similarly to my Software Development chapter, doing my Testing chapter alongside the development process would have been a big help. However, I feel that the main components/functionalities of my system have been tested here, which overall is the most important part of the project.

**Conclusion**

I have learned a lot about different methods of programming in Python during this project, such as using Singletons, MVC approach, using Controllers and Services to run SQL queries etc. Obviously, not all my goals for the system were achieved, but I think that with more time and experience I would be able to redo this project at a higher standard than before. Given that this is an A level project, and I am a 1-person team, the system operates quite robustly. I enjoyed the process of being met with a problem, learning of way to fix that problem, and then implementing these fixes in my own way. Overall, I feel that my coding skills have greatly improved, and I look forward to starting more projects in the future!